Eurographics Symposium on Parallel Graphics and Visualization (2021)
M. Hadwiger, M. Larsen, F. Sadlo (Editors)

Scalable In Situ Computation of Lagrangian Representations via Local Flow Maps

Sudhanshu Sane[1], Abhishek Yenpure[2], Roxana Bujack[3], Matthew Larsen[4], Kenneth Moreland[5], Christoph Garth[6], Chris R. Johnson[1] and Hank Childs[2]

[1]SCI Institute at University of Utah, [2]University of Oregon, [3]Los Alamos National Laboratory, [4]Lawrence Livermore National Laboratory, [5]Oak Ridge National Laboratory, [6]Technische Universität Kaiserslautern

Abstract

In situ computation of Lagrangian flow maps to enable post hoc time-varying vector field analysis has recently become an active area of research. However, the current literature is largely limited to theoretical settings and lacks a solution to address scalability of the technique in distributed memory. To improve scalability, we propose and evaluate the benefits and limitations of a simple, yet novel, performance optimization. Our proposed optimization is a communication-free model resulting in local Lagrangian flow maps, requiring no message passing or synchronization between processes, intrinsically improving scalability, and thereby reducing overall execution time and alleviating the encumbrance placed on simulation codes from communication overheads. To evaluate our approach, we computed Lagrangian flow maps for four time-varying simulation vector fields and investigated how execution time and reconstruction accuracy are impacted by the number of GPUs per compute node, the total number of compute nodes, particles per rank, and storage intervals. Our study consisted of experiments computing Lagrangian flow maps with up to 67M particle trajectories over 500 cycles and used as many as 2048 GPUs across 512 compute nodes. In all, our study contributes an evaluation of a communication-free model as well as a scalability study of computing distributed Lagrangian flow maps at scale using in situ infrastructure on a modern supercomputer.

CCS Concepts

• ***Human-centered computing*** → ***Scientific visualization;***

1. Introduction

As compute capabilities continue to outpace I/O capacity on supercomputers, *in situ* processing is an increasingly important solution to enable analysis of large-scale simulation data [BAA*16]. In situ processing involves coupling with the simulation code and operating on the full spatiotemporal resolution of the data in memory. However, in situ analysis tasks operate in constrained environments and are afforded limited execution time and memory. Consequently, analysis tasks must scale effectively since simulations execute across hundreds of compute nodes (CNs). In this paper, we investigate the performance of in situ data reduction via Lagrangian analysis to enable exploratory time-varying vector field analysis.

Lagrangian analysis is a powerful tool to explore time-varying vector fields generated by simulations. The notion of calculating a Lagrangian flow map, i.e., sets of particle trajectories, for an ocean modeling simulation “online" for “offline” exploration was first proposed by Vries et al. [VD01] two decades ago. More recently, compared to the traditional Eulerian technique under sparse temporal settings, Agranovsky et al. [ACG*14] evaluated reduced Lagrangian representations of time-varying vector fields and showed significantly improved accuracy-storage propositions for exploration. Figure 1 illustrates the approach. The Lagrangian flow maps

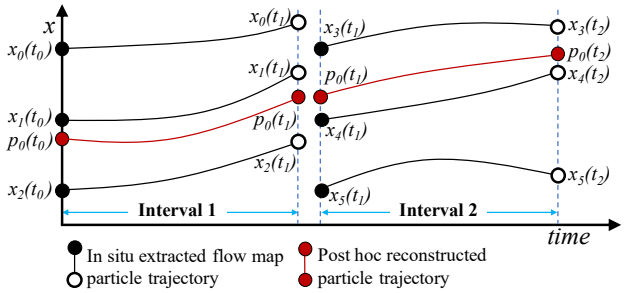


Figure 1: *The phases of Lagrangian analysis. The in situ phase uses uniform seed placement and extracts flow maps over temporally nonoverlapping intervals. In this example, the flow maps for the intervals $[t_0, t_1]$ and $[t_1, t_2]$ consist of particles $\{x_0, x_1, x_2\}$ and $\{x_3, x_4, x_5\}$, respectively. The extracted flow maps are used as input during the post hoc phase. Here, the trajectory of particle p_0 is calculated by interpolating the flow maps, i.e., from x_1 and x_2 in the first time interval, and x_3 and x_4 in the second time interval.*

are computed in situ using every cycle of a simulation, i.e., the full temporal resolution. After calculating and storing the flow maps, post hoc analysis tasks can use the flow maps to interpolate new particle trajectories to explore the vector field.

Table 1: *Differences in distributed-memory particle advection factors for in situ data reduction and post hoc analysis.*

Use Case	In situ reduction via Lagrangian flow map	Post hoc flow analysis
Input / Output	Simulation vector field / Flow map	Eulerian mesh or flow map / Trajectories
Objective	Store representation of vector field	Analysis and visualization of vector field
Vector Field Type	Unsteady (time-varying) state	Steady and unsteady state
Number of Particles	Depends on sampling strategy and data set size	Depends on analysis task and data set size
Number of Steps	Depends on storage interval/sampling strategy	Depends on analysis task
Memory Constraints	Shared with simulation code and typically limited	Can use all available memory
Spatial Decomposition and Data Access	Simulation-determined with each rank accessing one data block	Can be strategically modified, duplicated, requested on-demand, etc.
Communication	Required every cycle to compute a complete flow map	Can be strategically performed and/or delayed
Load Balance	Depends on sampling strategy	Depends on analysis task
Preprocessing	No prior work	Used in several works
Domain Coverage	Strategy to maintain domain coverage is necessary	Depends on analysis task

The Agranovsky et al. [ACG*14] study has been followed by several works to advance our understanding of the Lagrangian paradigm. However, the scalability of the technique while operating in situ has not been previously addressed. The authors of a recent comprehensive review of Lagrangian analysis [v*18], have identified the challenges of (1) utilization of heterogenous computer architectures, (2) providing parallel performance and scalability, and (3) lack of an accessible API that allows integration with different simulations. In this paper, we address the challenge of scalability, as well as utilize a runtime in situ infrastructure for multi-physics HPC simulations and GPUs for Lagrangian flow map computation.

With this study, our contributions include:

- A scalability study of distributed-memory particle advection using GPUs for Lagrangian flow map computation.
- An evaluation of a proposed performance optimization, i.e., computing local Lagrangian flow maps, across multiple extraction parameter configurations.

2. Background and Related Work

In the following section, we restrict our attention to the use of Lagrangian analysis as an in situ data reduction strategy and the relevant research on distributed-memory particle advection.

Lagrangian Analysis. In the Lagrangian specification of a time-varying vector field, information is stored using integral curves, where each curve encodes the trajectory of a single massless particle. Each integral curve provides insight regarding flow behavior in the vicinity of the particle's trajectory [BJ15]. Collectively, a large number of integral curves spanning the spatial domain can be defined in terms of a flow map, i.e., a Lagrangian representation of the flow field. The flow map $F_{t_0}^{t}(x_0) : \mathbb{R}^d \times \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}^d$ describes to where a particle starting at position $x_0 \in \mathbb{R}^d$ and time $t_0 \in \mathbb{R}$ moves in the time interval $[t_0, t] \subset \mathbb{R}$ [GGTH07].

Agranovsky et al. [ACG*14] proposed a two phase approach: (1) use in situ processing to compute reduced Lagrangian flow maps, and (2) interpolate the flow maps for time-varying vector field exploration. The study demonstrated significantly improved accuracy-storage propositions for exploration under sparse temporal settings compared to the traditional Eulerian approach. The Eulerian approach is susceptible to error due to high numerical approximation in settings of temporal sparsity [VB04, QvSSG14, ACG*14, SBC18, RLG19]. In comparison, the Lagrangian approach benefits from access to the complete spatiotemporal resolution to compute flow maps. Multiple works have advanced this research area by considering in situ sampling or extraction strategies [SCB19, RPD19], post hoc reconstruction and theoretical/empirical error analysis using the flow map as input [HSW11, AOGJ15, BJ15, COJ15, CBJ16, HBJG16, SBC18, SCB19, RPD19, JGG20], and the use of reduced data sets for analysis of ocean modeling applications [NBSS17].

In this paper, we focus on the first phase, i.e., in situ computation of a Lagrangian flow map. Agranovsky et al. [ACG*14] considered a strategy prioritizing domain coverage. Particles were seeded along a uniform grid, interpolated for a fixed number of cycles, i.e., *storage interval*, and reset to repeat the process until the end of the simulation. Their study used CPUs on as many as 128 CNs. Two other works have proposed relevant sampling strategies. First, Sane et al. [SCB19] proposed tracing longer trajectories, while storing intermediate particle locations at fixed storage intervals. In this approach, as particles diverged and clustered, a reconstruction error-guided sampling strategy added or removed particles. Second, Rapp et al. [RPD19] applied a statistical sampling strategy based on the blue noise property to the extraction of particle trajectories. Although these strategies can yield an improved reconstruction compared to uniform or random sampling, they are limited to a single CN, and they require additional research to keep execution time and/or memory requirements acceptable in a distributed-memory setting. Execution time and memory are limited resources when operating in situ, so the task of efficiently computing a Lagrangian flow map in distributed-memory remains challenging.

Distributed-Memory Particle Advection. Distributed-memory particle advection algorithms have historically been proposed for post hoc analysis. We refer readers to comprehensive surveys [PRN*11, ZY18] on this topic for steady and unsteady vector fields in multinode settings. However, the existing post hoc solutions are not directly applicable due to the different constraints and requirements of computing a Lagrangian flow map in situ. We identify and list these differences in Table 1.

The most significant challenge associated with distributed-memory particle advection is scalability. Poor scalability is due to the frequent and large amount of communication required between processors as particles continue trajectory integration across rank

domain boundaries. Most studies in this area have required particle exchange, but a few have used preprocessing or runtime operations to address scalability by limiting communication. Notably, Chen et al. [CXLS11, CS13] and Liao et al. [LMKK19] modify the data layout as a preprocessing step to propose parallelize-over-data techniques. Using parallelize-over-particles strategies, Guo et al. [GHS*14] and Zhang et al. [ZGY16] moved and duplicated data between processors or used on-demand data loading. Lastly, as an advection acceleration strategy, Bleile et al. [BSGC17] computed block-specific flow maps for a single time-slice as a communication-free preprocess and used the mapping to transport particles across entire blocks in a single step. Considering in situ constraints, and the time-varying nature of the data, existing strategies are either not applicable or viable.

In this paper, we consider a communication-free model for extracting local Lagrangian flow maps. We compare this approach with the traditional computation of a Lagrangian flow map, which does incur communication costs. Our comparisons are at scale and evaluate the respective performance benefits and reconstruction accuracy of the approach. Based on our understanding of existing research in this area, in situ constraints, and practical temporal storage intervals, we believe the communication-free model is viable for a sampling strategy, and most importantly, intrinsically enables scalability to a large number of processors.

3. Methods

This section contains four parts. Section 3.1 describes the in situ infrastructure for our approaches. The next two sections describe the two Lagrangian flow map extraction approaches: the traditional approach (3.2) and a new communication-free approach (3.3). Section 3.4 describes how post hoc reconstruction, which applies to both approaches, is performed. Figures 3a, 3b, 3c, and 3d provide notional examples to compare both approaches. Further, based on the in situ system classification in [C*20], the Lagrangian flow map extraction system is classified as one with a dedicated API integration, on-node proximity, direct access, a time division of execution, automatic operation, and a derived output type.

3.1. In Situ Infrastructure

Our study used the Ascent in situ infrastructure [LAA*17]. The Ascent API can be used to integrate with a simulation code and access various in situ analytics capabilities. It can also be used to create a workflow when loading data sets from disk. The fundamental operation of particle advection required to compute the particle trajectories that form the Lagrangian flow map is implemented using the VTK-m library [MSU*16]. VTK-m is a platform-portable scientific visualization library for shared-memory parallel environments. This library enabled us to easily engage GPUs for particle advection. Specifically, in situ particle advection is performed using fourth-order Runge-Kutta for interpolation. Ascent has VTK-h [LAA*17] as a distributed-memory wrapper around VTK-m. VTK-h uses MPI and acts as the communication layer. Figure 2 provides an illustration of the workflow for in situ data reduction. Ascent is invoked every cycle of the simulation, and it consequently invoked the relevant calls to the Lagrangian *filter* (VTK-h + VTK-m). The rank-specific Lagrangian filter used the simulation

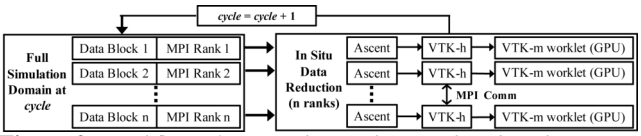


Figure 2: *Workflow of in situ data reduction distributed across n processes to compute a Lagrangian flow map.*

vector field data to advect particles every cycle, and triggered the storage of trajectories that comprise the Lagrangian flow map. Lagrangian analysis modules are available in the latest releases of the open-source Ascent [LAA*17] and VTK-m [MSU*16] libraries.

3.2. Traditional Computation of Lagrangian Flow Maps

Conceptually, a Lagrangian representation encodes the behavior of a time-varying flow using particle trajectories. Given the early nature of investigations into the use of Lagrangian flow maps, the best ways to seed, represent in memory, communicate, and store flow maps are open questions. Section 2 noted three prior sampling strategies; however, none of these were evaluated at scale. As a baseline, our work implemented the technique described by Agranovsky et al. [ACG*14] and is referred to as Lagrangian$_{Dist}$.

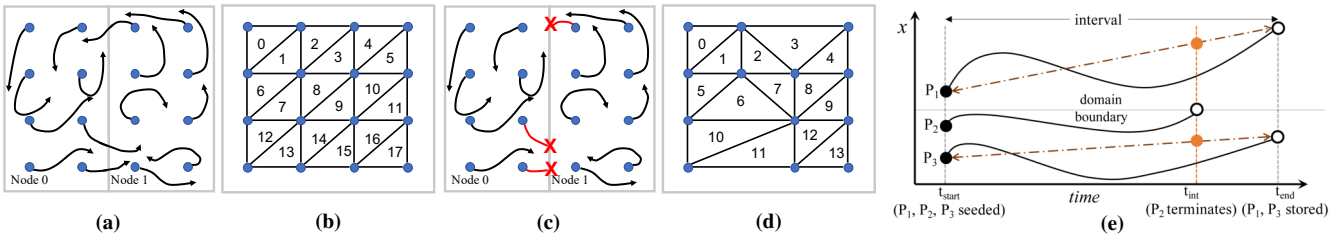
For Lagrangian$_{Dist}$, we defined the overall strategy for flow map computation by considering four aspects:

- **Sampling:** For each data block, seeds are placed along a uniform grid. As particles are advected, they may diverge or cluster. To maintain domain coverage, particles are terminated after a fixed number of cycles, i.e., storage interval, and their end locations are saved, followed by a new set of particles uniformly seeded at the original starting locations. Thus, sets of temporally non-overlapping particle trajectories are calculated that span the full duration of the simulation.
- **In-memory Representation:** Adopting a minimal approach, only the current location and the validity of the particle are statically stored in memory.
- **Communication:** Ranks communicated every cycle to check for particles (incoming/outgoing) crossing rank boundaries to continue trajectory integration. Control is returned to the simulation after all communication is completed.
- **Storage:** Agranovsky et al. did not specify how particles that exited the domain before the end of the storage interval are saved, which is problematic for post hoc reconstruction. In our implementation, we stored the end location and a Boolean indicating validity, i.e., whether the particle remained within the domain for the entirety of the storage interval.

3.3. Computation of Local Lagrangian Flow Maps

With this work, we propose an optimization to address the increasing cost of communication as the scale increases. Our strategy is a simple, yet novel and scalable, approach: skip all the communication and compute only local Lagrangian flow maps in situ. We refer to this implementation as Lagrangian$_{Local}$ and define the overall strategy as follows:

- **Sampling:** Similar to Lagrangian$_{Dist}$, we used a uniform seed placement and a fixed storage interval. Additionally, seeds can be placed redundantly along domain boundaries in adjacent ranks, and although our study does not consider ghost zones, we believe these would serve to strengthen our proposed optimization.



(a) (b) (c) (d) (e)

Figure 3: *Figures 3a and 3c notionally illustrate the in situ computation of Lagrangian$_{Dist}$ and Lagrangian$_{Local}$ flow maps, respectively. Figures 3b and 3d show the corresponding global Delaunay triangulations (see Section 3.4 for details of post hoc reconstruction). Figure 3e illustrates a case of early termination, which is discussed in Section 3.3.*

- **In-memory representation:** Similar to Lagrangian$_{Dist}$, we stored the current location and validity of a particle.
- **Communication:** We eliminated all particle information exchange and synchronization. Particles that required communication to continue trajectory integration were discarded. Thus, all ranks operated independently.
- **Storage:** For a uniform grid, we stored the end location (3 double) and validity (1 Boolean). Particles that were discarded were marked as invalid.

An interesting consideration was whether to store the particle termination time and boundary location. Figure 3e illustrates the problem with this approach. Three particle trajectories (P_1, P_2, P_3) start at time t_{start}. Two of the particle trajectories (P_1, P_3) remain in their domain until t_{end}, i.e., for the entire storage interval, and one particle (P_2) reaches the domain boundary at t_{int}. During post hoc reconstruction, using the information of P_2 to transport a new particle from t_{start} to t_{int} requires knowing the location of neighboring particles (P_1, P_3) at t_{int}. Only the particle start and end locations are known in our case, so this approach would require linearly interpolating (depicted using the dotted brown line) the locations of the neighboring particles at t_{int}. However, these interpolated locations are often erroneous (orange particles).

The benefits of computing local Lagrangian flow maps are reduced execution time and improved scalability characteristics. Further, for storage intervals that are used in practice, we hypothesize only a small percentage of particles will be discarded. However, the loss of information in the form of discarded particle trajectories could reduce the quality of flow reconstruction. Our study evaluates this trade-off. Lastly, we believe this performance optimization could be applied adaptively, i.e., communication can be turned on/off when appropriate. Our study assumed an "always off" approach, which we view as invaluable for future adaptive designs.

3.4. Post Hoc Flow Field Reconstruction

We used a Lagrangian-based advection scheme similar to prior work [ACG*14, BJ15]. New particles interpolate temporally nonoverlapping sets of particle trajectories. For a specific storage interval, the starting positions of all valid trajectories are treated as points in an unstructured mesh. To calculate the trajectory of a particle P, the *neighborhood*, i.e., the containing cell of the unstructured mesh, is first identified. We computed the cells of the unstructured mesh by performing Delaunay triangulation. The identified neighborhood forms a convex hull around the particle P. Next, P "follows" the neighborhood until the end of the specific storage interval. To compute the next location of P, we used the end locations of the neighborhood particles and barycentric coordinate interpolation. This process can continue for several steps, with each advection step advancing the particle forward in time by the length of the storage interval.

4. Theoretical Error Analysis

Although our main contribution is an empirical evaluation, for generality, this section provides a theoretical error analysis of our strategy to compute local Lagrangian flow maps.

We used a one-dimensional linear interpolation L of a function $f : \mathbb{R} \to \mathbb{R}$ for $x \in [x_0, x_1] \subset \mathbb{R}$

$$L_{f(x_0),f(x_1)}(x) = \frac{x - x_0}{x_1 - x_0} f(x_1) + \frac{x_1 - x}{x_1 - x_0} f(x_0). \tag{1}$$

Higher dimensional results satisfy (1) for each component.

In our approach, a particle starting at x_1 that reached the node boundary before a storage cycle is discarded. Thus, its function value (the flow map) is not known. However, we can reconstruct it from its known neighbors $L_{f(x_0),f(x_2)}(x_1)$. Consider a particle $x \in [x_0, x_1] \subset \mathbb{R}$ whose path is interpolated using this reconstructed value. We get the same result as if we had used the closest existing neighbors directly. Let L' denote $L_{f(x_0),L_{f(x_0),f(x_2)}(x_1)}(x)$, then

$$\begin{aligned} L' &\overset{(1)}{=} \frac{x - x_0}{x_1 - x_0} L_{f(x_0),f(x_2)}(x_1) + \frac{x_1 - x}{x_1 - x_0} f(x_0) \\ &\overset{(1)}{=} \frac{x - x_0}{x_1 - x_0} \left(\frac{x_1 - x_0}{x_2 - x_0} f(x_2) + \frac{x_2 - x_1}{x_2 - x_0} f(x_0) \right) + \frac{x_1 - x}{x_1 - x_0} f(x_0) \\ &= \frac{x - x_0}{x_2 - x_0} f(x_2) + \frac{(x_2 - x_1)(x - x_0) + (x_2 - x_0)(x_1 - x)}{(x_2 - x_0)(x_1 - x_0)} f(x_0) \\ &= \frac{x - x_0}{x_2 - x_0} f(x_2) + \frac{x_2 - x}{x_2 - x_0} f(x_0) \\ &\overset{(1)}{=} L_{f(x_0),f(x_2)}(x). \end{aligned} \tag{2}$$

Bujack et al. [BJ15] previously established that the post hoc interpolation of pathlines is a numerical one-step integration method. Its accuracy is bounded by its global truncation error at stitching step $n \in \mathbb{N}$ of

$$e_n \leq \frac{d^2}{8} (t_n - t_0) h_x^2 \max_{\tau \in [t_0, t_n]} \max_{\zeta \in \mathbb{R}^d} \| H_{\dot{F}^\tau_{t_{j-1}}}(\zeta) \|_\infty e^{L(t_n - t_0)} \tag{3}$$

with dimension d, start time t_0, end time t_n, spatial Lipschitz constant L, Hessian H of the temporal derivative of the flow map $\dot{F}$, and spatial distance h_x between the flow map trajectories. A detailed derivation of equation (3) can be found in [BJ15]. As a result, the interpolation error is $O(h_x^2)$ if the flow map has bounded second derivatives in space and first derivatives in time, which is a

reasonable assumption for a differentiable vector field, because the solutions of an initial value problem depend smoothly on the initial conditions and time [Har02].

Equation (2) shows that the error bound $O(\tilde{h}_x^2)$ still holds but with a larger $\tilde{h}_x > h_x$. Its size is determined by the size of missing information, which is limited by the maximum distance particles can move in one time interval. If a particle is seeded further than $\max_{x_i \in \mathbb{R}^d} \max_{j=1..n} \|F_{j-1}^{j}(x_i)\|$ away from the node boundary, it cannot reach it and must therefore have the correct flow map information. In the worst case, we can have missing information on both sides of the boundary, and therefore it follows from the mean value theorem that

$$\begin{aligned} \tilde{h}_x &\leq 2h_x + 2 \max_{x_i \in \mathbb{R}^d} \max_{j=1..n} \|F_{j-1}^{j}(x_i) - x_i\| \\ &\leq 2h_x + 2h_t \max_{x \in \mathbb{R}^d} \max_{t \in [t_{j-1}, t_j]} \|v(x,t)\| \end{aligned} \tag{4}$$

with the underlying velocity field $v : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ and the temporal step size h_t, which is the interval time between storing data to disk, usually around one-thousandth of the total integration time. The future increase of the global truncation error of a particle that traverses this region can continue even after it has left the region, but for all particles that never enter the region close to the boundary, the original error bound holds.

5. Study Overview

This section describes our data sets (5.1), experiments (5.2), evaluation considerations (5.3), and runtime environment (5.4).

5.1. Data Sets

We computed Lagrangian flow maps for four data sets.

Cloverleaf3D. This mini ECP application solves compressible Euler equations in a hydrodynamics setting on a Cartesian grid using an explicit second-order method [MBG*13]. Cloverleaf3D has been used in several studies to evaluate emerging architectures and various techniques targeting large-scale applications. The simulation is initially relatively stable and begins with an energy bar expanding from the center. Configurations for the Cloverleaf3D simulation varied from 64^3 ($\approx$ 262k cells) to 812^3 ($\approx$ 535M cells).

Arnold-Beltrami-Childress (ABC) Flow. This popular turbulent velocity field is a solution of Euler's equations in 3D for incompressible, inviscid fluid flows and is parameterized using three variables (we use $A = B = C = 1$). We considered a grid size of 256^3 ($\approx$ 16.5M cells) and generated 400 cycles of a time-dependent variant [BCT01] with a time step of 0.001.

Nyx. The Nyx simulation is a N-body and gas dynamics code for large-scale cosmological simulations [ABL*13]. We used a test executable named TurbForce and generated 400 cycles of a 128^3 data set ($\approx$ 2M cells) with a time step of 0.002.

Jet Flow. This data set is a simulation of a jet of high-velocity fluid entering a medium at rest. It was created using the Gerris Flow Solver [Pop03]. The vector field is defined over a $128 \times 256 \times 128$ grid ($\approx$ 4.1M cells) and 300 cycles with a step size of 0.001.

5.2. Experiment Setup

For this study, we conducted experiments in two phases.

Phase-I. The first phase of the experiments was conducted by directly integrating Ascent with the Cloverleaf3D simulation code. We conducted a weak scaling study on a modern supercomputer and considered the impact of the number of ranks and GPUs engaged per CN on execution time. Additionally, we performed two comparisons referred to as **C-I** and **C-II** using the Cloverleaf3D simulation code:

- **C-I.** We compared Lagrangian$_{Local}$ to Lagrangian$_{Dist}$ when the number of ranks for a fixed grid size, i.e., the degree of domain decomposition, varied.
- **C-II.** We compared Lagrangian$_{Local}$ to the Eulerian technique to verify accuracy-storage benefits observed in prior works remain.

The results of **Phase-I** are presented in Section 6.1.

Phase-II. For the second phase of experiments, we worked with the ABC, Nyx, and Jet time-varying data sets. We created a workflow using Ascent and loaded files from disk for every cycle. A fixed number of resources (1 GPU/rank, 4 ranks/CN, 16 CNs) were used for these experiments. Here, the number of particles used to sample the domain was controlled by a data reduction factor (denoted by 1:X, i.e., one particle for every X grid points). We also considered multiple values for the storage interval. These tests provided insight into reconstruction accuracy, especially as vector field and configuration parameters varied, as well as computational performance. The results of **Phase-II** are presented in Section 6.2.

5.3. Evaluation

Execution Time. We measured different types of execution time — particle advection, communication, and total time — and reported the average per cycle across the entire run. To simplify comparisons, we excluded cycles at the end of a storage interval. These cycles included a communication cost incurred by Lagrangian$_{Dist}$ to return all particles to the respective origin nodes. In this paper, we do not analyze the parallel I/O times because (1) I/O times on supercomputers can be highly variable and lead to difficult interpretations, (2) infrequently performed I/O write times for our tests were less than the corresponding advection cost for a single step, and (3) local flow maps consistently led to fewer bytes stored, meaning we would expect I/O costs to not increase.

For post hoc reconstruction, a Delaunay triangulation was performed using CGAL [CGA20] on a single-node workstation. In our experiments, the number of points for both Lagrangian$_{Local}$ and Lagrangian$_{Dist}$ were of the same magnitude, and thus, required similar reconstruction times.

Reconstruction Accuracy. We quantitatively measured accuracy in two contexts: (1) reconstruction of any discarded particle trajectories and (2) full pathlines. In both cases, we measured Euclidean distances and represented error as a percentage of the grid cell side (GCS) to provide a perspective relative to the simulation domain. An error under 100% of GCS indicates a particle trajectory end point within one grid cell side of the ground truth.

The first priority was understanding how accurately we could reconstruct the flow map. Since Lagrangian$_{Local}$ discards particle

trajectories, potentially leaving a void of samples in the domain near boundaries, we reconstructed these trajectories by interpolating the stored flow map. We compared the reconstructed particle trajectories to the ground truth by measuring the Euclidean distance between the end points. The statistics do not include all the particle trajectories of the flow map that were successfully computed and stored since these are already accurate. Including these trajectories would skew the overall result. Therefore, for each test, the percentages of both stored (accurate) and discarded (reconstruction error measured) particle trajectories are presented. The measured reconstruction error is presented using violin plots and heatmaps (2D histograms) to emphasize the distribution of error.

Second, for **C-I** and **C-II** we measured the accuracy of new pathlines. In this case, pathlines were computed for the entire duration of the simulation run, and we reported the average L2-norm over all interpolated locations for each particle.

5.4. Runtime Environment

We tested the Lagrangian analysis techniques by running the experiments on Summit (a supercomputer at ORNL). Each CN of Summit has two IBM Power9 CPUs, each with enhanced on-chip acceleration via NVLink to 3 GPUs; the total GPUs per CN is 6. Each GPU is a NVIDIA Tesla V100 with 5120 CUDA cores and 16 GBytes of HBM2 memory.

6. Results

We organize our results into two subsections, with each subsection focusing on results from a phase of experiments.

6.1. Phase-I Results Using Cloverleaf3D Simulation Code

We report the results of **Phase-I** experiments that include a weak scaling study, **C-I**, and **C-II**.

6.1.1. Weak Scaling Study

We considered 17 configurations of the Cloverleaf3D simulation in our weak scaling **Phase-I** study. Our experiment configurations span 81^3 cells across 2 MPI ranks on 1 CN to 812^3 cells across 2048 MPI ranks on 512 CNs, with each MPI rank operating on an approximately 64^3 grid. In each case, one GPU was assigned to every MPI rank, and we varied the number of ranks per CN. With respect to the number of MPI ranks on each CN, we ran 1 test using 2 ranks, 10 tests using 4 ranks, and 6 tests using 6 ranks. For each run, we terminated the simulation after 500 cycles. Given the variation in grid size, each test reached a different stage of the simulation. As the simulation grid size increased, the step size used by the simulation decreased. With respect to parameterization of the Lagrangian flow map computation, we used a storage interval of 25 cycles and a data reduction of 1:8 ($\approx 32,768$ particles seeded per MPI rank) for every test.

Execution Time. Figure 4 compares the average total time required per step by Lagrangian$_{Dist}$ and Lagrangian$_{Local}$. As the scale of the simulation increased, the cost of communication dominated the execution time for Lagrangian$_{Dist}$. In comparison, Lagrangian$_{Local}$ scaled better because each MPI rank operated independently. For the range of configurations considered, Lagrangian$_{Local}$ demonstrated up to 4x speed-up over

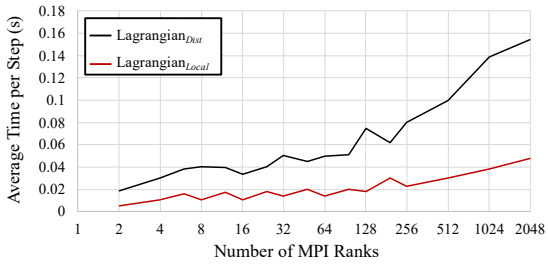
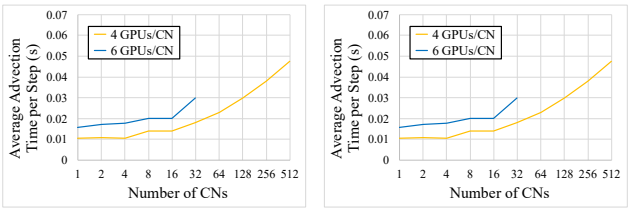


Figure 4: *Results of the weak scaling study on the Cloverleaf3D data set, which shows the poor scalability of Lagrangian$_{Dist}$ as the number of MPI ranks increases.*

Lagrangian$_{Dist}$. Further, we expect this speed-up would increase even more as the scale of the simulation increases. The cost of particle advection (for both techniques) increased as the scale of the simulation increased. However, this increase in particle advection cost was attributed to the difference in each test domain and the RK4 kernel that performed velocity field interpolation. Further, use of a faster particle advection kernel would result in greater speed-ups for Lagrangian$_{Local}$.

The "sawtooth" nature of the line curves in Figure 4 is the result of a series of test configurations alternating between 4 and 6 ranks per CN. Varying the number of ranks (each using 1 GPU) on each CN impacted both particle advection and communication costs. Figure 5 isolates these costs to show the weak scaling trends. From Figure 5a, particle advection performed better with 4 GPUs per CN versus 6. Use of shared memory by multiple GPUs on a single CN and saturation of the NVLink by the VTK-m particle advection kernel caused this effect. In contrast, the MPI communication cost showed a reduction when using 6 ranks versus 4 per CN as seen in Figure 5b. On-node MPI communication optimizations contributed to better performance when grouping a larger number of MPI ranks on each CN. However, the communication cost of inter-node remained high in comparison to intra-node. Configurations using 4 ranks per CN showed particularly poor scalability as the number of CNs increased.

Reconstruction Accuracy. We calculated the reconstruction accuracy for 8 of the Lagrangian$_{Local}$ weak scaling **Phase-I** configurations, which we labeled T1 through T8 (see Figure 6). Config-



(a) *Costs of particle advection.* **(b)** *Costs of communication.*

Figure 5: *Results for weak scaling the number of GPUs or ranks (4 or 6) per compute node. Lagrangian$_{Local}$ particle advection costs are plotted in 5a and Lagrangian$_{Dist}$ communication costs are in 5b. While particle advection performed better with fewer GPUs sharing memory on a single compute node (5a), communication benefitted from MPI optimizations when more ranks execute on a single compute node (5b).*

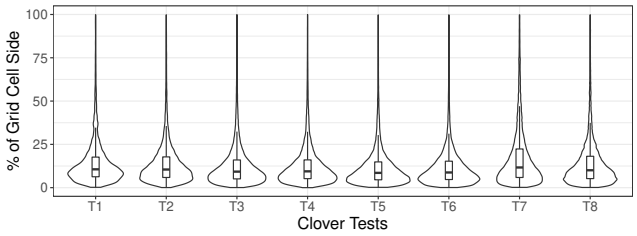


Figure 6: *The distribution of the reconstruction error for eight Cloverleaf3D tests (labeled T1 through T8). The grid dimensions increase l-r from* 81^3 *to* 204^3 *over 500 cycles.*

urations terminated 2%-5% of particles and consequently stored 95%-98% of all initially seeded trajectories. We measured the accuracy of reconstructing the discarded trajectories only and present the results using violin plots in Figure 6. As the simulation grid resolution increased, the grid cell size and particle advection step size decreased, while the total number of particles sampling the domain increased. Over 75% of discarded trajectories have reconstruction error under 25% of a GCS from the ground truth. Further, this trend was observed across all the 8 test configurations we reconstructed.

Overall, using Lagrangian$_{Local}$ with a storage interval of 25 cycles and 1:8 data reduction factor, the complete Lagrangian flow map was reconstructed accurately (under 100% of GCS) while providing speed-ups of 2x-4x for the Cloverleaf3D data set.

6.1.2. C-I Results

Multiphysics HPC simulations typically have millions of grid points per rank and increase grid resolution as the number of ranks increases. To identify limitations, we evaluated Lagrangian$_{Local}$ in situations where this was not the case, i.e., when the number of processes operating over a fixed grid size increased.

Since the Lagrangian$_{Local}$ strategy was to discard trajectories exiting the rank-specific domain, this approach was susceptible to low-resolution data blocks (i.e, sampling would use a small number of particles per rank) and longer storage intervals (i.e., integration times). This experiment measured the error of 1,000 new pathlines generated for 800 cycles compared to the ground truth. Further, three parameter options were considered for domain decomposition, storage interval, and data reduction factor. Figure 7

Fixed Grid Size: 64^3		Degree of Domain Decomposition								
		1 Block			8 Blocks			32 Blocks		
Reduction Factors →		1	8	27	1	8	27	1	8	27
Lagrangian$_{Local}$	25	5	11	24	5	12	24	5	12	33
	50	5	10	21	5	11	21	7	14	28
	100	5	10	18	5	11	19	12	20	35
	Interval									
Reduction Factors →		1	8	27	1	8	27	1	8	27
Lagrangian$_{Dist}$	25	5	11	24	5	12	24	5	12	24
	50	5	10	21	5	11	21	5	12	21
	100	5	10	18	5	10	18	5	11	19
	Interval									

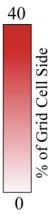


Figure 7: *Average error of new pathlines traced using Lagrangian$_{Dist}$ and Lagrangian$_{Local}$ for varying domain decomposition, storage interval, and data reduction factor. Here, a* 64^3 *Cloverleaf3D data set defined over 800 cycles is used. Accuracy measurements are compared to the ground truth.*

shows the pathline error when interpolating the flow maps generated by Lagrangian$_{Dist}$ and Lagrangian$_{Local}$. The accuracy of Lagrangian$_{Local}$ remained close to Lagrangian$_{Dist}$ until the domain decomposition was at its highest (64^3 grid decomposed across 32 ranks). Although the overall accuracy of the interpolated pathlines was high (within a single GCS on average for all tests), both techniques lost some accuracy as the storage interval and data reduction factor increased, with Lagrangian$_{Local}$ performing worse under greater domain decomposition. These results highlight the limitations of the Lagrangian$_{Local}$ technique as is.

6.1.3. C-II Results

Here, we compared Lagrangian$_{Local}$ to an Eulerian representation with temporal subsampling. Table 2 shows the results of **C-II**. We considered three storage intervals: 20, 40, and 60. We used 96 MPI ranks distributed across 16 CNs and a grid size of 586^3. Lagrangian$_{Local}$ used a data reduction of 1:8, whereas for the Eulerian technique we stored the full spatial resolution. To compare accuracy, we reconstructed 100,000 randomly seeded pathlines for 600 cycles. Overall, Lagrangian$_{Local}$ was increasingly accurate (6x to 11x) compared to the Eulerian approach as the interval size increased, but required less data storage. These results aligned with findings in prior works [ACG*14, SBC18] that compared the use of Lagrangian representations to the traditional approach under sparse temporal settings.

Table 2: *Comparison of Lagrangian$_{Local}$ and the traditional approach, i.e., the Eulerian representation with temporal subsampling, for the Cloverleaf3D dataset. The error is the average percentage of grid cell side and is computed for 100,000 pathlines over 600 cycles. The unit for storage is GB.*

Storage Interval	Lagrangian$_{Local}$		Eulerian	
	Avg % of GCS	Storage	Avg % of GCS	Storage
20	18.8	34	115.4	267
40	25.2	17	269.0	133
60	37.5	12	424.8	95

6.2. Phase-II Results Using ABC, Nyx, Jet Data Sets

This section presents results for three time-varying data sets. We considered a fixed amount of compute resources and varied configuration parameters that impact both execution time and reconstruction accuracy. Figure 8 contains the computation costs for all **Phase-II** experiments. Additionally, we derived, visualized, and compared the finite-time Lyapunov exponent (FTLE) scalar field using select Lagrangian$_{Local}$ flow maps for each of these data sets to the ground truth FTLE field.

6.2.1. ABC Flow

Table 3 lists all the ABC **Phase-II** test configurations.

Execution Time. Using 64 MPI ranks across 16 CNs, each with 1 GPU, Lagrangian$_{Local}$ required up to 2.8x less execution time as Lagrangian$_{Dist}$. Figure 8 shows particle advection and communication costs are directly proportional to the particle count. Further, particle advection for up to 2M particles (32k particles per GPU) was performed in under 0.004 seconds per step for all data sets.

Reconstruction Accuracy. Figure 9's violin plots indicate the

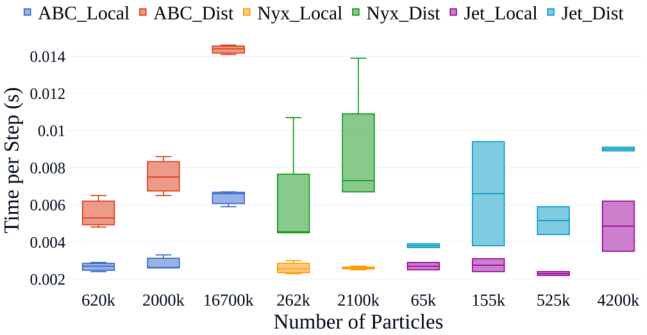


Figure 8: *Execution time per step of Lagrangian$_{Dist}$ and Lagrangian$_{Local}$ grouped by data set and ordered by the number of particles for **Phase-II** experiments. Overall, most Lagrangian$_{Local}$ tests required under 0.005 seconds per step and showed small variance. In contrast, Lagrangian$_{Dist}$ tests showed higher cost per step as well as more variability.*

Table 3: *Specifications for 9 ABC tests. In most cases, over 90% of the complete flow map was stored.*

Test	Interval	Reduction	Particles	Stored	Discarded
T1	25	1:1	16700k	96.9%	3.1%
T2	50			94.3%	5.7%
T3	100			89.7%	10.3%
T4	25	1:8	2098k	97.7%	2.3%
T5	50			95.2%	4.8%
T6	100			90.4%	9.6%
T7	25	1:27	621k	98.6%	1.4%
T8	50			95.9%	4.1%
T9	100			91.8%	8.2%

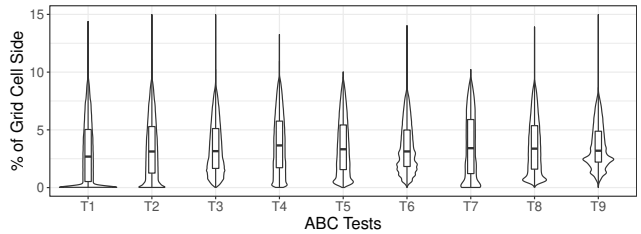
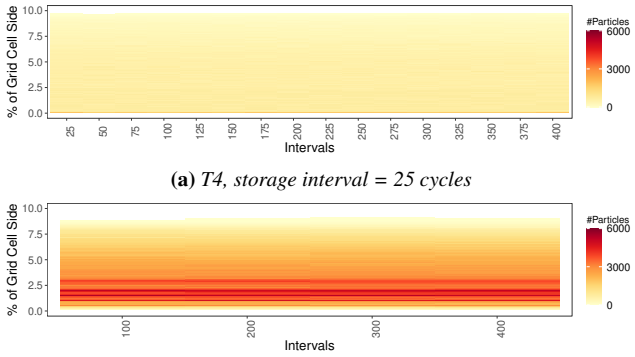


Figure 9: *Violin plots of reconstruction error for the ABC data set tests labeled T1-T9. The plots show the error remained within a consistent range across tests, with only small increases as the data reduction factor and storage interval increase.*

ABC data set showed shifts in reconstruction error distribution when varying the storage intervals and sampling resolution. The error range for all configurations, however, was similar. The discarded trajectories from the ABC data set Lagrangian flow map were reconstructed accurately, with trajectories interpolated to the same grid cell as the ground truth. Specifically, for all 9 tests, the mean reconstruction error was between 2.5% and 5% of GCS from the ground truth. Figure 11 visualizes the FTLE field derived from the reduced Lagrangian$_{Local}$ test T6. Although reconstruction near extrema is similar to the ground truth, the FTLE is overestimated in other regions and artifacts are visible along node boundaries.

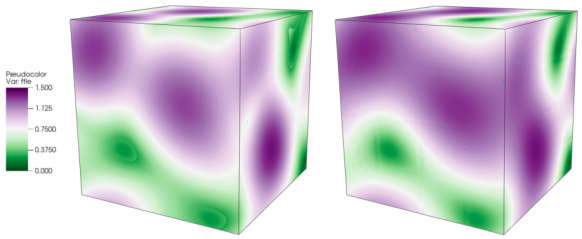
Impact of Varying Storage Interval. Figure 10 shows heatmaps of the reconstruction error across all intervals for tests T4 and T6. For a straightforward comparison, the figures show the absolute



(a) *T4, storage interval = 25 cycles*

(b) *T6, storage interval = 100 cycles*

Figure 10: *Heatmap of reconstruction error for the ABC data set as a function of interval. These plots show that the majority of reconstructed trajectories are within a grid cell side of the ground truth, despite the larger of number of discarded particles resulting from the larger storage interval (10b).*



(a) Ground truth, 100 cycles (b) T6, 1:8, Interval = 100

Figure 11: *3D colormapped surfaces of the FTLE field for the ABC data set. Although extrema of the FTLE field derived using the reduced Lagrangian$_{Local}$ flow map (right) are preserved, other regions are overestimated.*

number of particles reconstructed for each storage interval. Comparing these heatmaps, the effect of increasing the storage interval was that a larger number of particles were terminated in a single interval, and thus reconstruction error increased.

6.2.2. Nyx Cosmology

Table 4 lists all the Nyx **Phase-II** test configurations.

Table 4: *Specifications for 8 Nyx tests. Nyx tests had the largest percentage of discarded particles among all our experiments.*

Test	Interval	Reduction	Particles	Stored	Discarded
T1	10	1:1	2097k	96%	4%
T2	20			92.1%	7.9%
T3	40			85.7%	14.3%
T4	50			83%	17%
T5	10	1:8	262k	97.4%	2.6%
T6	20			93.7%	6.3%
T7	40			88.2%	11.8%
T8	50			85.8%	14.2%

Execution Time. Across our 8 tests using the Nyx data set, Lagrangian$_{Local}$ computed a flow map up to 5.2x faster than the corresponding Lagrangian$_{Dist}$ flow map. Additionally, Figure 8 shows the standard deviation was greater for the Lagrangian$_{Dist}$ tests and was caused by the larger number of particles exchanges between ranks for this data set.

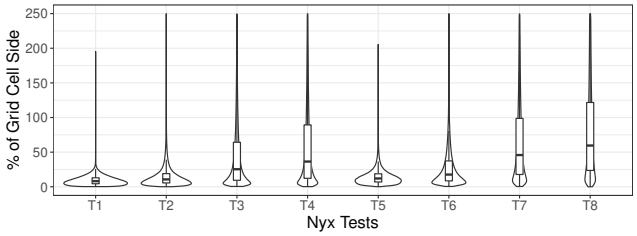
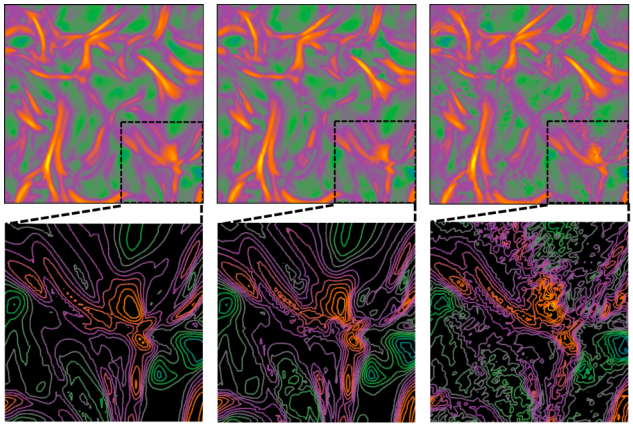


Figure 12: *Violin plots of reconstruction error for the Nyx data set tests labeled T1-T8. For most tests (T1-T6), the majority of particles can be reconstructed within a grid cell side of the ground truth. However, for tests with increased storage interval and data reduction (T7, T8) we found the error can be higher.*



(a) Ground truth, 40 cycles (b) T3, 1:1, Interval = 40 (1 interval interpolated) (c) T1, 1:1, Interval = 10 (4 intervals interpolated)

Figure 13: *2D colormapped slices (top row) and isolines of a subset region (bottom row) of the FTLE field for the Nyx data set. Here, we visualize the impact of propagating error via interpolation of consecutive flow maps. Using the flow maps of the Lagrangian$_{Local}$ T3 test, a single interval is interpolated and the reconstructed FTLE field is similar to the ground truth. Using the flow maps of T1, we interpolated 4 intervals to calculate the FTLE over 40 cycles. In this case, although the slice visualization shows the overall structure well, the isolines reveal the noise introduced locally by "stitching" Lagrangian$_{Local}$ flow maps together.*

Reconstruction Accuracy. As a consequence of a large number of particles being discarded by Lagrangian$_{Local}$, the accuracy of reconstruction was impacted for tests with longer storage intervals. 7 of 8 tests showed up to the third quartile reconstructing under 100% of a GCS. Comparing tests T3 and T4 (1:1 data reduction factor) to tests T7 and T8 (1:8 data reduction factor), although T3 and T4 discarded a greater percentage of samples, they remained more accurate due to the absolute number of particles used to sample the domain. For shorter storage interval lengths (T1, T2, T5, T6), the reconstruction quality was high for both data reduction factors. Figure 13 compares the FTLE field derived by interpolating a single longer interval (T3) to interpolating multiple intervals (T1) of Lagrangian$_{Local}$ flow maps. Interpolating multiple intervals leads to error propagation. The consequence of the error propagation was the noise introduced in the FTLE derived using T1 flow maps. In contrast, interpolating the single longer interval of T3 flow maps generated an FTLE field closer to the ground truth.

6.2.3. Jet Flow

Table 5 lists all the Jet **Phase-II** test configurations.

Table 5: *Specifications for 8 Jet tests. In most cases, under 2% of particle trajectories are discarded. Accurate reconstruction of these trajectories (see Figure 14) was dependent on the absolute number of stored particles.*

Test	Interval	Reduction	Particles	Stored	Discarded
T1	5	1:1	4194k	99.4%	0.6%
T2	10			97.9%	2.1%
T3	5	1:8	524k	99.6%	0.4%
T4	10			98.9%	1.1%
T5	5	1:27	155k	99.7%	0.3%
T6	10			99.1%	0.9%
T7	5	1:64	65k	99.8%	0.2%
T8	10			99.3%	0.7%

Execution Time. For the Jet data set, Lagrangian$_{Local}$ computed a flow map up to 3.9x faster than the corresponding Lagrangian$_{Dist}$ flow map. For this data set, we considered shorter storage intervals, and thus a smaller percentage of particles required particle exchange to continue trajectory integration. However, Figure 8 shows the variability in the cost of communication as it was susceptible to network usage and bandwidth contention. A single outlier Lagrangian$_{Local}$ test showed a higher computation cost.

Reconstruction Accuracy. The Jet data set presented an adversarial case for our proposed optimization of computing local flow maps. The data set contained regions with high velocity magnitude. Across the range of configurations in Figure 14, both the storage interval and the data reduction factor impacted the reconstruction accuracy. 6 of 9 tests had a mean reconstruction error under 100% of GCS. Further investigation into the distribution of T2 revealed that the longer storage interval and 1:1 sampling resulted in several particles terminating in easy-to-reconstruct areas of the domain at the start of the simulation (most other configurations showed particle termination only after cycle 30). For tests using a storage interval of 10 (T2, T4, T6, T8), the reconstruction accuracy is reduced.

Impact of Varying Data Reduction Factor. Figure 15 compares the distribution of reconstruction error as the data reduction factor increases. Each cell in the heatmap shows a percentage of particles to enable comparison between tests. For test T1 using a 1:1 sampling, reconstruction accuracy was high and a larger percentage of particles were reconstructed with error under 100% of GCS.

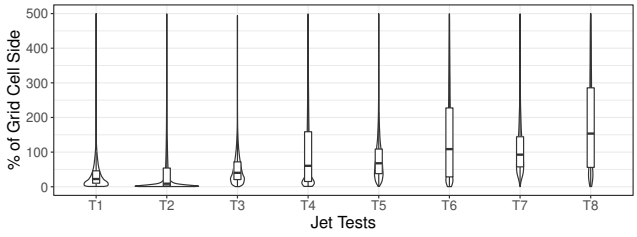
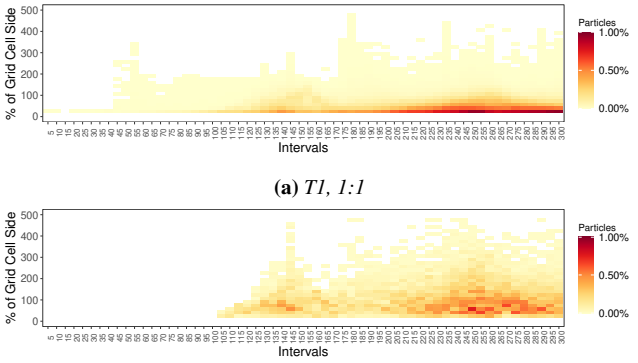


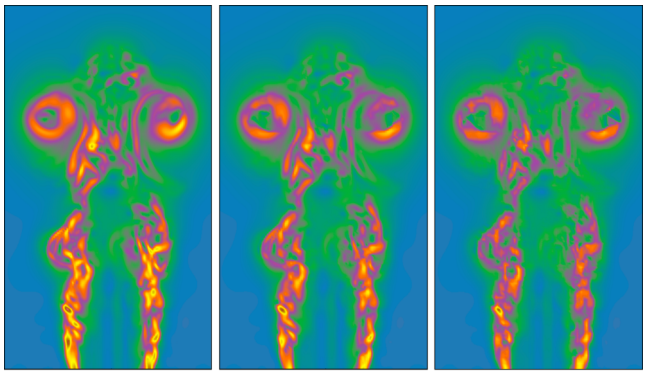
Figure 14: *Violin plots of reconstruction error for the Jet data set tests labeled T1-T8. The plots show that error increased significantly with increased data reduction factor and storage interval. Reconstruction error was low only when no data reduction (1:1) was used (T1, T2).*



(a) *T1, 1:1*

(b) *T7, 1:64*

Figure 15: *Heatmaps of reconstruction error for the Jet data set. These plots highlight the low tolerance for data reduction for this data set — a high data reduction factor resulted in an increase in the percentage of particles reconstructed up to a few grid cell sides away from the ground truth.*



(a) Ground truth (b) T1, 1:1 (c) T3, 1:8

Figure 16: *2D colormapped slices of the FTLE field computed over 5 cycles for the Jet data set. In both cases (b, c), flow maps over a single interval of Lagrangian$_{Local}$ are interpolated.*

As the number of particles used to sample the domain reduced, a higher percentage of particles were reconstructed multiple cells away from the ground truth. In Figure 15b, test T7 used a 1:64 data reduction factor and there are more red/orange cells with higher error during the later stages of the simulation. Figure 16 visualizes an FTLE field derived using the T1 (1:1) and T3 (1:8) Lagrangian$_{Local}$ flow maps. Although for both tests the overall FTLE ridge structure can be visualized, for T3 using fewer particles, the loss of accuracy due to discarded trajectories is more evident.

7. Limitations and Discussion

While scalability is the major benefit of local Lagrangian flow maps, this benefit comes with certain limitations as shown in our evaluation. While both Lagrangian$_{Dist}$ and Lagrangian$_{Local}$ show error (as shown in Figure 7), the error of Lagrangian$_{Local}$ increases faster if the storage interval is too large or if too few sampled are stored. Lastly, although our strategy relies on spatiotemporal coherance, discarding a large number of particles could introduce biases in the extracted flow map and misrepresent regions.

We briefly discuss two solutions to these challenges. The first potential solution is adapt execution to adopt some local communication with spatially adjacent ranks. The second solution is to stop discarding trajectories that hit the domain boundary, and instead utilize them for reconstruction. In this case, Lagrangian analysis could remain communication-free but would need enhancements not only for how to represent trajectories (termination location and time, number of points stored along pathline), but also would require a novel post hoc interpolation scheme. For example, extensions of recent works on in situ sampling strategies [SCB19, RPD19] and post hoc reconstruction leveraging deep learning [JGG20] offer potential solutions.

Another challenge for Lagrangian analysis is adaptively determining the appropriate storage interval. Additionally, motivated by our results and to limit uncertainty, we believe it would be worthwhile to investigate flow visualization techniques that utilize individual intervals of flow maps. Lastly, our paper presented a study that considered the entire flow field without an analysis of specific features. Research understanding the uncertainty of specific features of interest introduced by various reduced representations (Eulerian, Lagrangian, ZFP, etc.) would be valuable to the visualization community. We believe these challenges should be pursued for future work. That said, we believe a number of use cases for Lagrangian analysis in practice today could benefit from flow maps extracted using a communication-free model.

8. Conclusion

Lagrangian analysis is increasingly being considered as a solution to reduce memory footprint while providing high accuracy for time-varying vector fields under sparse temporal settings. In this paper, we presented a study computing in situ Lagrangian flow maps using GPUs at scale on a modern supercomputer with a focus on scalability. We proposed the computation and use of local Lagrangian flow maps and evaluated the benefits and limitations of a simple strategy using multiple time-varying vector field data sets. Our study evaluated the impact of the number of particles and test configuration parameters (GPUs, rank, compute nodes) on the total, particle advection, and communication time during in situ analysis. The largest configuration we considered computed 67M particle trajectories over 500 cycles of a 812^3 grid using 2048 GPUs across 512 compute nodes. We empirically showed that for several practical configurations local Lagrangian flow maps can be interpolated with a minimal loss of accuracy, while requiring a fraction of the execution time. Finally, the computation of local Lagrangian flow maps demonstrated scalability and offered greater predictability for in situ analysis costs.

Acknowledgment

The authors acknowledge research support provided by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. DOE Office of Science and National Nuclear Security Administration, the Intel Graphics and Visualization Institutes of XeLLENCE, the NIH under grant numbers P41 GM103545 and R24 GM136986, and the DOE under grant number DE-FE0031880. This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the U.S. DOE Office of Science under Contract No. DE-AC05-00OR22725.

References

[ABL*13] ALMGREN A. S., BELL J. B., LIJEWSKI M. J., LUKIĆ Z., VAN ANDEL E.: Nyx: A massively parallel amr code for computational cosmology. *The Astrophysical Journal 765* (2013), 39. 5

[ACG*14] AGRANOVSKY A., CAMP D., GARTH C., BETHEL E. W., JOY K. I., CHILDS H.: Improved post hoc flow analysis via lagrangian representations. In *4th IEEE Symposium on Large Data Analysis and Visualization, LDAV* (2014), pp. 67–75. 1, 2, 3, 4, 7

[AOGJ15] AGRANOVSKY A., OBERMAIER H., GARTH C., JOY K. I.: A multi-resolution interpolation scheme for pathline based lagrangian flow representations. In *Visualization and Data Analysis* (2015), vol. 9397, International Society for Optics and Photonics. 2

[BAA*16] BAUER A. C., ABBASI H., AHRENS J., CHILDS H., GEVECI B., KLASKY S., MORELAND K., O'LEARY P., VISHWANATH V., WHITLOCK B., ET AL.: In situ methods, infrastructures, and applications on high performance computing platforms. In *Computer Graphics Forum* (2016), vol. 35, Wiley Online Library, pp. 577–597. 1

[BCT01] BRUMMELL N., CATTANEO F., TOBIAS S.: Linear and nonlinear dynamo properties of time-dependent abc flows. *Fluid Dynamics Research 28*, 4 (2001), 237–265. 5

[BJ15] BUJACK R., JOY K. I.: Lagrangian representations of flow fields with parameter curves. In *5th IEEE Symposium on Large Data Analysis and Visualization, LDAV* (2015), pp. 41–48. 2, 4

[BSGC17] BLEILE R., SUGIYAMA L., GARTH C., CHILDS H.: Accelerating advection via approximate block exterior flow maps. *Electronic Imaging 2017*, 1 (2017), 140–148. 3

[C*20] CHILDS H., ET AL.: A Terminology for In Situ Visualization and Analysis Systems. *International Journal of High Performance Computing Applications 34*, 6 (2020), 676–691. 3

[CBJ16] CHANDLER J., BUJACK R., JOY K. I.: Analysis of error in interpolation-based pathline tracing. In *18th Eurographics Conference on Visualization, EuroVis - Short Papers* (2016), pp. 1–5. 2

[CGA20] CGAL PROJECT: *CGAL User and Reference Manual*, 5.1 ed. CGAL Editorial Board, 2020. 5

[COJ15] CHANDLER J., OBERMAIER H., JOY K. I.: Interpolation-based pathline tracing in particle-based flow visualization. *IEEE Transactions on Visualization and Computer Graphics 21*, 1 (2015), 68–80. 2

[CS13] CHEN C. M., SHEN H. W.: Graph-based seed scheduling for out-of-core FTLE and pathline computation. In *3rd IEEE Symposium on Large-Scale Data Analysis and Visualization, LDAV* (2013), pp. 15–23. 3

[CXLS11] CHEN C., XU L., LEE T., SHEN H.: A flow-guided file layout for out-of-core streamline computation. In *IEEE Symposium on Large Data Analysis and Visualization, LDAV* (2011), pp. 115–116. 3

[GGTH07] GARTH C., GERHARDT F., TRICOCHE X., HANS H.: Efficient computation and visualization of coherent structures in fluid flow applications. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1464–1471. 2

[GHS*14] GUO H., HONG F., SHU Q., ZHANG J., HUANG J., YUAN X.: Scalable lagrangian-based attribute space projection for multivariate unsteady flow data. In *IEEE Pacific Visualization Symposium, PacificVis* (2014), pp. 33–40. 3

[Har02] HARTMAN P.: *Ordinary Differential Equations*. Society for Industrial and Applied Mathematics, 2002. 4

[HBJG16] HUMMEL M., BUJACK R., JOY K. I., GARTH C.: Error estimates for lagrangian flow field representations. In *18th Eurographics Conference on Visualization, EuroVis - Short Papers* (2016), pp. 7–11. 2

[HSW11] HLAWATSCH M., SADLO F., WEISKOPF D.: Hierarchical line integration. *IEEE Transactions on Visualization and Computer Graphics 17*, 8 (2011), 1148–1163. 2

[JGG20] JAKOB J., GROSS M., GUNTHER T.: A fluid flow data set for machine learning and its application to neural flow map interpolation. *IEEE Transactions on Visualization and Computer Graphics*, 01 (2020), 1–1. 2, 10

[LAA*17] LARSEN M., AHRENS J. P., AYACHIT U., BRUGGER E., CHILDS H., GEVECI B., HARRISON C.: The ALPINE in situ infrastructure: Ascending from the ashes of strawman. In *Proceedings of the In Situ Infrastructures on Enabling Extreme-Scale Analysis and Visualization, ISAV@SC* (2017), pp. 42–46. 3

[LMKK19] LIAO Y., MATSUI H., KREYLOS O., KELLOGG L. H.: Scalable parallel flow visualization using 3d line integral convolution for large scale unstructured simulation data. In *19th Eurographics Symposium on Parallel Graphics and Visualization, EGPGV@EuroVis* (2019), pp. 17–25. 3

[MBG*13] MALLINSON A., BECKINGSALE D. A., GAUDIN W., HERDMAN J., LEVESQUE J., JARVIS S. A.: Cloverleaf: Preparing hydrodynamics codes for exascale. *The Cray User Group* (2013). 5

[MSU*16] MORELAND K., SEWELL C., USHER W., LO L.-T., MEREDITH J., PUGMIRE D., KRESS J., SCHROOTS H., MA K.-L., CHILDS H., ET AL.: Vtk-m: Accelerating the visualization toolkit for massively threaded architectures. *IEEE Computer Graphics and Applications 36*, 3 (2016), 48–58. 3

[NBSS17] NARDINI P., BÖTTINGER M., SCHEUERMANN G., SCHMIDT M.: Visual study of the benguela upwelling system using pathline predicates. In *5th Workshop on Visualisation in Environmental Sciences, EnvirVis@EuroVis* (2017), pp. 19–23. 2

[Pop03] POPINET S.: Gerris: a tree-based adaptive solver for the incompressible euler equations in complex geometries. *Journal of Computational Physics 190*, 2 (2003), 572–600. 5

[PRN*11] PETERKA T., ROSS R. B., NOUANESENGSY B., LEE T., SHEN H., KENDALL W., HUANG J.: A study of parallel particle tracing for steady-state and time-varying flow fields. In *25th IEEE International Symposium on Parallel and Distributed Processing, IPDPS* (2011), pp. 580–591. 2

[QvSSG14] QIN X., VAN SEBILLE E., SEN GUPTA A.: Quantification of errors induced by temporal resolution on lagrangian particles in an eddy-resolving model. *Ocean Modelling 76* (2014), 20–30. 2

[RLG19] ROCKWOOD M. P., LOISELLE T., GREEN M. A.: Practical concerns of implementing a finite-time lyapunov exponent analysis with under-resolved data. *Experiments in Fluids 60*, 4 (2019), 1–16. 2

[RPD19] RAPP T., PETERS C., DACHSBACHER C.: Void-and-cluster sampling of large scattered data and trajectories. *IEEE Transactions on Visualization and Computer Graphics 26*, 1 (2019), 780–789. 2, 10

[SBC18] SANE S., BUJACK R., CHILDS H.: Revisiting the evaluation of in situ lagrangian analysis. In *18th Eurographics Symposium on Parallel Graphics and Visualization, EGPGV@EuroVis* (2018), pp. 63–67. 2, 7

[SCB19] SANE S., CHILDS H., BUJACK R.: An interpolation scheme for VDVP lagrangian basis flows. In *19th Eurographics Symposium on Parallel Graphics and Visualization, EGPGV@EuroVis* (2019), pp. 109–119. 2, 10

[v*18] VAN SEBILLE E., ET AL.: Lagrangian ocean analysis: Fundamentals and practices. *Ocean Modelling 121* (2018), 49 – 75. 2

[VB04] VALDIVIESO DA COSTA M., BLANKE B.: Lagrangian methods for flow climatologies and trajectory error assessment. *Ocean Modelling 6*, 3 (2004), 335–358. 2

[VD01] VRIES P., DÖÖS K.: Calculating lagrangian trajectories using time-dependent velocity fields. *Journal of Atmospheric and Oceanic Technology 18*, 6 (2001), 1092–1101. 1

[ZGY16] ZHANG J., GUO H., YUAN X.: Efficient unsteady flow visualization with high-order access dependencies. In *IEEE Pacific Visualization Symposium, PacificVis* (2016), pp. 80–87. 3

[ZY18] ZHANG J., YUAN X.: A survey of parallel particle tracing algorithms in flow visualization. *Journal of Visualization 21*, 3 (2018), 351–368. 2